Assignment - 16

Task 1

Write a simple program to show inheritance in scala.

Task 2

Write a simple program to show multiple inheritance in scala

Task 3

Write a partial function to add three numbers in which one number is constant and two numbers can be passed as inputs and define another method which can take the partial function as input and squares the result.

Task 4

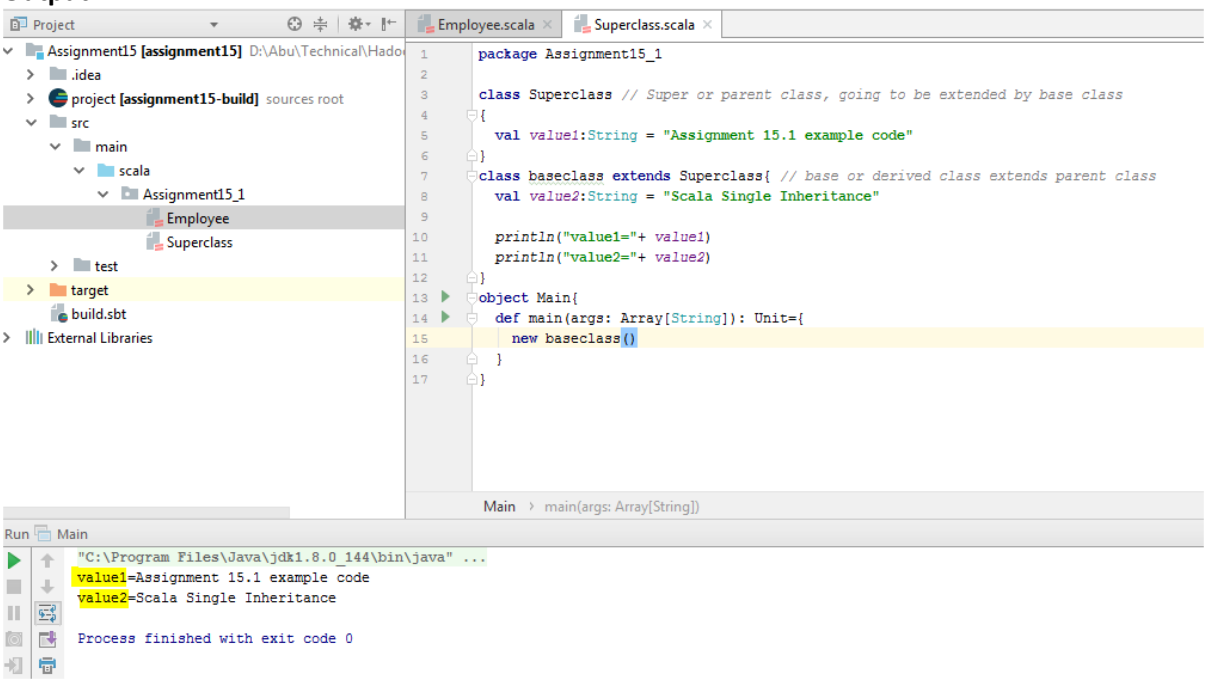
Write a program to print the prices of 4 courses of Acadgild:
Android App Development -14,999 INR
Data Science - 49,999 INR
Big Data Hadoop & Spark Developer – 24,999 INR
Blockchain Certification – 49,999 INR
using match and add a default condition if the user enters any other course.

Task 1

Write a simple program to show inheritance in scala.

```
package Assignment15_1

class Superclass // Super or parent class, going to be extended by base class
{
  val value1:String = "Assignment 15.1 example code"
}
class baseclass extends Superclass{ // base or derived class extends parent class
  val value2:String = "Scala Single Inheritance"

  println("value1="+ value1)
  println("value2="+ value2)
}
object Main{
  def main(args: Array[String]): Unit={
    new baseclass()
  }
}
```

Output



Task 2

Write a simple program to show multiple inheritance in scala

```
package Assignment15_1

trait MultipleInheritance //parent trait
{
  def show() // defining the function show()
  {
    println("Assignment 15.1")
  }
}
trait one extends MultipleInheritance // extending the parent trait
{
  override def show()
  {
    println("This won't be printed")
  }
}
trait two extends MultipleInheritance // extending the parent trait
{
  override def show()
  {
    println("Acadgild Scala Multiple Inheritance Example")
  }
}
class three extends one with two  //extending the base traits, calling the function
show()

object MainMulti{
  def main(args:Array[String]): Unit ={
    var c:three = new three // it will call last function which is mentioned in the
class three, changing the order will give different result
    c.show()
  }
}
```

Output

```scala
package Assignment15_1

trait MultipleInheritance
{
  def show()
  {
    println("Assignment 15.1")
  }
}
trait one extends MultipleInheritance
{
  override def show()
  {
    println("This won't be printed")
  }
}
trait two extends MultipleInheritance
{
  override def show()
  {
    println("Acadgild Scala Multiple Inheritance Example")
  }
}
class three extends one with two

object MainMulti{
  def main(args:Array[String]): Unit ={
    var c:three = new three
    c.show()
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Acadgild Scala Multiple Inheritance Example

Process finished with exit code 0
```

```scala
package Assignment15_1

trait MultipleInheritance
{
  def show()
  {
    println("Assignment 15.1")
  }
}
trait one extends MultipleInheritance
{
  override def show()
  {
    println("This won't be printed")
  }
}
trait two extends MultipleInheritance
{
  override def show()
  {
    println("Acadgild Scala Multiple Inheritance Example")
  }
}
class three extends two with one

object MainMulti{
  def main(args:Array[String]): Unit ={
    var c:three = new three
    c.show()
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
This won't be printed

Process finished with exit code 0
```

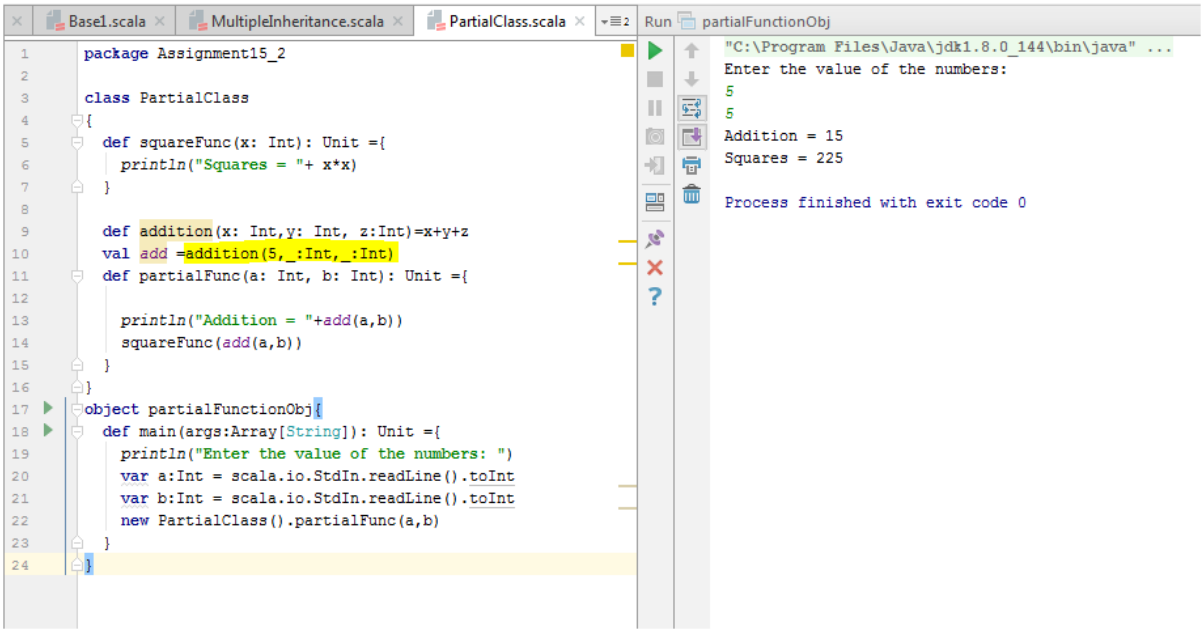
Task 3

Write a partial function to add three numbers in which one number is constant and two numbers can be passed as inputs and define another method which can take the partial function as input and squares the result.

```
package Assignment15_2

class PartialClass
{
  def squareFunc(x: Int): Unit ={
    println("Squares = "+ x*x) // defined a function to square the input's
  }

  def addition(x: Int,y: Int, z:Int)=x+y+z//a function to add constant+value1+value2
  val add =addition(5,_:Int,_:Int) // the constant value = 5
  def partialFunc(a: Int, b: Int): Unit ={ // another method to define a value for
constant


    println("Addition = "+add(a,b))
    squareFunc(add(a,b))
  }
}
object partialFunctionObj{ // singleton object to call the functions
  def main(args:Array[String]): Unit ={
    println("Enter the value of the numbers: ")
    var a:Int = scala.io.StdIn.readLine().toInt // reading the input value
    var b:Int = scala.io.StdIn.readLine().toInt
    new PartialClass().partialFunc(a,b) //
  }
}
```

Here the constant is x and we defined the value of **x as 5**, we have two variables a and b, we pass **a=y=5** and **b=z=5**, we get the **x+y+z** = **5+5+5 = 15**.
15 is the output of the partial function is squared **15*15** in the **squareFunc** which is **225**.

Output

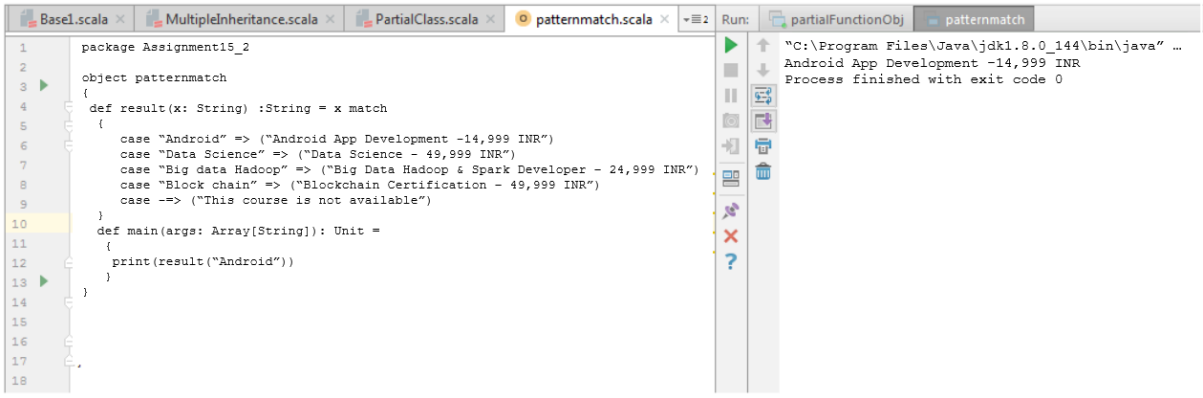


Task 4

Write a program to print the prices of 4 courses of Acadgild:
Android App Development -14,999 INR
Data Science - 49,999 INR
Big Data Hadoop & Spark Developer – 24,999 INR
Blockchain Certification – 49,999 INR
using match and add a default condition if the user enters any other course.

```
package Assignment15_2
object patternmatch
{
 def result(x: String) :String = x match
  {
     case "Android" => ("Android App Development -14,999 INR")
     case "Data Science" => ("Data Science - 49,999 INR")
     case "Big data Hadoop" => ("Big Data Hadoop & Spark Developer – 24,999 INR")
     case "Block chain" => ("Blockchain Certification – 49,999 INR")
     case => ("This course is not available")
  }
  def main(args: Array[String]): Unit =
   {
    print(result("Big Data Hadoop & Spark Developer"))
   }
}
```

Output



Base1.scala | MultipleInheritance.scala | PartialClass.scala | patternmatch.scala

```scala
package Assignment15_2

object patternmatch
{
 def result(x: String) :String = x match
  {
     case "Android" => ("Android App Development -14,999 INR")
     case "Data Science" => ("Data Science - 49,999 INR")
     case "Big data Hadoop" => ("Big Data Hadoop & Spark Developer - 24,999 INR")
     case "Block chain" => ("Blockchain Certification - 49,999 INR")
     case -=> ("This course is not available")
  }
  def main(args: Array[String]): Unit =
   {
    print(result("Core Java"))
   }
}
```

Run: partialFunctionObj | patternmatch

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" …
This course is not available
Process finished with exit code 0
```